[Editorial] Special issue on computational thinking and coding in childhood

Article (Accepted Version)

Editorial - Special Issue on Computational Thinking and Coding in Childhood

Kate Howland[1], Judith Good[1], Judy Robertson[2], Andrew Manches[2]

[1]Department of Informatics, University of Sussex, {k.l.howland@sussex.ac.uk; j.good@sussex.ac.uk}
[2]Moray House School of Education, University of Edinburgh, {judy.robertson@ed.ac.uk; a.manches@ed.ac.uk}

This Special Issue on Computational Thinking and Coding in Childhood brings together ten articles addressing the topic from a range of perspectives and disciplinary backgrounds. Amid increased international interest in younger children's engagement with computer science and programming, many questions remain about how to provide tools, environments and curricula that embody appropriate progression and engender motivation for these topics across the years. The papers in this volume address and discuss a number of these questions, as well as raising important new questions.

The idea for the Special Issue was prompted by our 2015 Interaction Design and Children Workshop "Every Child a Coder? Research Challenges for a 5-18 Programming Curriculum" [1]. A key workshop activity involved participants with expertise in computer science, education, and developmental psychology collaborating on sketching a roadmap for computational thinking development. This prompted interesting discussions about how key concepts, such as abstraction, problem decomposition, states and data structures, might map to abilities and motivations at different developmental stages. Our conversations suggested the need for a spiral curriculum, where knowledge develops and deepens over time with repeated revisiting of concepts using relevant examples. Workshop participants also highlighted the importance of assessment methods and developmentally appropriate tools and activities, and emphasised the benefits of drawing on educational theory in related subjects long taught to younger children, such as mathematics.

The papers in this Special Issue include valuable contributions on the assessment of computational thinking ability, methods for understanding children's attitudes towards learning programming and investigations of social interaction in collaborative programming activities. There are also important contributions in the form of classroom-based evaluations of novel methods, activities and tools, including a robotics programme, a network-based environment, a plug and play toolkit for physical computing, game-based learning, and explorations of interdisciplinary approaches through mathematics and literature projects.

Clearly, it is important to remember and draw upon lessons learnt from efforts to integrate computing education into classrooms in the past, where one key theme has been the importance of teacher knowledge and confidence. This issue is addressed directly in Chalmers' paper on robotics and computational thinking in primary schools [2]. The paper focuses on the perceptions and experiences of teachers in Australia across primary level in using robotics to teach computational thinking, and includes teachers working with the youngest age group, which is important to gain a more comprehensive picture of the challenges of integrating computing education across school. The study reported worked with a relatively small sample, only four teachers, yet was able to triangulate findings from questionnaires, journal reflections and interviews to provide valuable insight into the

teachers' experiences. The work revealed the value of robotic kits in helping teachers build their confidence and knowledge whilst providing a timely reminder of the need to supplement such pedagogical tools with well-designed professional development.

An advantage of approaching computing education through computational thinking is that it is suitable for interdisciplinary projects. de Paula, Burn, Noss and Valente [3] explore how computational thinking and arts and humanities learning can complement each other in an informal learning context at the British Library. The paper analyses an interactive narrative version of the epic Anglo-Saxon poem Beowulf which was created by two teenagers using the Mission Maker game authoring environment, to show how programming can be a compelling bridge between computational thinking and arts and humanities. This is an insightful example of how young people can engage with culturally important stories in a way which is appropriate to them, but putting it in a meaningful computational context.

A longstanding question in computing education research is the relationship between what is learnt when programming and learning in other areas – the issue of transfer. Benton, Saunders, Kalas, Hoyles and Noss [4] remind us that we should refrain from simply expecting domain transfer "as a spin-off from learning to program" but rather something we might facilitate by developing the right environments, activities and pedagogical support. In their paper, the authors report on their efforts to achieve this as part of a two year intervention project, ScratchMaths (SM), which seeks to exploit programming for the learning of mathematics.

Benton et al.'s paper shares the experiences of two classes using a mathematical algorithm for addition expressed through programming as a means to develop understanding of the place-value concept. Their analysis focused on both the extent to which pupils could successfully implement the programming functionality and whether their learning was subsequently exploited in the specified mathematical context, and the techniques employed by the teacher to scaffold transfer and the potential impact on pupils' engagement and expressions of place value. Their findings demonstrate the potential of well-designed activities, alongside teacher support, to engage pupils in difficult mathematical ideas, whilst highlighting the need for an iterative design research process to match learning content with progression in both computing and mathematics curriculum areas. Their paper ends with a timely reminder that we should take advantage of the new wave of enthusiasm for computing education not to learn program for its programming's sake but to ensure that "programming truly becomes a purposeful medium of expression within our education systems".

Papavlasopoulou, Sharma and Giannakos [5] present an eye-tracking study designed to investigate the links between gaze data and self-reported learning and attitudinal data in young people's coding activities. They highlight the importance of understanding children's attitudes to coding activities, emphasising the importance of motivation from a number of perspectives. Given that eye-tracking has proved a helpful method in understanding adult programmer's activities and cognitive mechanisms, the authors set out to examine whether this approach could reveal potential associations between children's attitudes and gaze in programming task. This question was investigated in the context of a constructivist inspired game-making workshop in which children aged 8-17 used Scratch to make games. Statistical analysis of the eye-tracking data with attitudinal survey data showed significant links between self-reported learning, intention and attitude and gaze data including average

fixation duration, change in saccade direction and saccade amplitude. The authors offer some interesting interpretations of what these relationships might mean, and draw helpful links with previous findings. A number of additional questions are also raised by the work, including how gender and age might influence mappings of this sort, and more detail on the nature of the link between positive motivation and low cognitive load.

As Weintrop and Wilensky note in their paper [6], there has been a recent increase in languages that blend the characteristics of blocks-based and text-based languages, some allowing the user to switch between blocks-based and text-based representations, or hybrid environments, while others incorporating the features of both in a single representation. These new languages have led to a number of studies which have tried to answer the question of which language is “better”, or “easier to learn”. In contrast, Weintrop and Wilensky take a more finely-grained approach, arguing, from a theoretical perspective, that tools and representations shape our practices (i.e. the ways in which we use them) as well as our conceptual understandings (i.e. what we learn as a result of using them).

In a study of 90 high school aged novice programmers learning programming using either a blocks-based, text-based or hybrid environment, Weintrop and Wilensky found indeed that the hybrid modality was not “better’ than the others. Instead they found that the hybrid modality led to what they call “summative behaviour”, where students use the affordances of both modalities, sometimes to their benefit (i.e. using blocks as a way of supporting the less taxing cognitive process of recognition, rather than recall) and sometimes to their detriment (i.e. introducing syntax errors through the use of the text-based modality). Weintrop and Wilensky urge that further research is needed in order to better understand the specific properties of these emerging interfaces and their relationship to learning.

Gomes, Falcão and Tedesco [7] looked at how we might lower the barriers to learning programming for young children. Specifically, the authors examined six popular digital games for teaching computing concepts to children aged 5-7. In a qualitative, action-research study, where the lead researcher was also the children’s teacher, the authors found that there were a certain number of concepts that could not adequately be understood by young children despite seeing the concept in each of the different games and, in some cases, seeing them represented in different ways across games. However, the authors also found that certain representations of the concepts were more helpful than others. Finally, the authors examined the effects of the interactional affordances of the games on children’s understanding of the concepts, and their associated behaviours. For example, they interestingly found that when children’s programs did not function as intended, they erased the entire program and started from scratch, rather than trying to debug the program. The authors suggest an incremental approach to debugging, where children can see the correct statements within their incorrect program, might provide support for finding and correcting errors in their programs. Taken together, the authors provide a number of suggestions for further improving the ways in which we represent computing concepts to young children, and the ways in which children can interact with these environments.

As computational thinking becomes a more established part of school curricula around the world, educators have turned their attention to how it may be assessed. It is a sign of a maturing field that we now have the CTt assessment which has be shown to be valid and reliable in large scale studies with Spanish learners. Continuing this work, Román-

González, Pérez-González, Moreno-León and Robles [8] report on a study which investigated the extent to which their CTt assessment can be used to 'talent spot' for learners who might be particularly gifted in computational thinking. Indeed, results of the study with 314 middle school learners showed that CTt scores can be used to distinguish between top computational thinkers and regular computational thinkers when these categories are defined by the pace at which the learners successfully progress from visual to textual languages in a self-paced online environment. In addition, CTt scores correlate with programming scores (as one might expect from the close relationship between these skill sets) and CTt scores also correlates with grades in Informatics, Maths and Language. The latter result is important as our field strives to understand the relationship between CT and other academic areas and cognitive processes. Although CT correlates with other academic abilities, it is to some extent distinct from them, and so we need to have assessment tools like CTt to help with high quality teaching and learning, as well as basic research.

Katterfeldt, Cukurova, Spikol and Cuartielles [9] present and evaluate a toolkit designed to help educators teach programming to 14-18 year olds using collaborative approaches and physical computing. The researchers grounded their iterative design of the toolkit in a sound understanding of the demands of real classrooms, and contribute a study in a school context, as well as further user studies in school, festival and university contexts. The plug-and-play Arduino hardware and linked visual programming language is well-justified in relation to the target learners, and a rich set of data was collected through different evaluation studies. A nuanced picture is presented in the findings, with positive results relating to motivation and usability, but negative feedback from high school students about the support for collaboration provided by the toolkit. The authors are able to reflect helpfully on the limitations of the toolkit as evaluated, and offer eight key suggestions for high school teachers implementing project-based physical computing activities, with a particularly focus improving students' motivation and collaboration skills.

Tsan et al. [10] provide a fascinating account of pair programming behaviours in learners aged 9-11. Although the authors acknowledge the potential benefits of collaboration in programming, they note that most research has focussed on adults and/or older children. In a study of children aged 9-11, the authors investigated the ways in which the pairs shared the roles of driver and navigator, as well as the overall "talk time" of each partner, and the specific nature of their contributions. However, even in cases where roles and contributions appear to be shared reasonably equitably, a further qualitative analysis suggested that the nature of the interaction was sometimes far from collaborative (e.g. each member of the pair spoke for an equal length of time, but were not coordinating their utterances in ways that helped them move forward with the task). The authors suggest that general collaboration skills, such as self-advocacy and strategic question-asking, may well improve the quality of children's collaboration in pair programming, and in turn, have a positive impact on their learning. Indeed, in a field which has typically suffered from a lack of diversity (in terms of gender, ethnicity, disability, etc.), learning explicit approaches which can foster positive collaboration amongst programmers of all ages may well have a broader role in promoting diversity and inclusion.

Kelly, Finch, Bolles and Shapiro [11] also present a physical computing programming environment, but their approach is specifically designed to allow young people to create their

own networked systems. The authors also contribute two evaluation studies, carried out in US middle schools with 12-14 year olds, which were designed in collaboration with teachers at the schools. The BlockyTalky environment, which is aimed at children aged 10 and over, supports block-based-programming of physical devices in a browser window, with each device running a webserver and the code for each device being stored and executed on that device. One of the most notable elements of the study findings was the wide variety of different types of creative projects created by students using the environment and kit, including many new musical interfaces and games, and the range of computer science concepts covered. The authors provide a rich analysis of two of the projects in greater detail as case studies, and point to some interesting observations about design features, as well as social interaction and collaboration. There are promising indications about the potential for such environments to support inclusive participation and simultaneous code authoring. The authors also highlight the need for more longitudinal research to further investigate and extend the initial findings, and raise interesting questions about how computational thinking manifests and can be assessed in a distributed computational context.

In considering the ten papers in this special issue as a whole, four key themes emerge. The first theme, understandably, centres on the interrelationship between representations, activities and environments, and on the ways in which they support novice programmers and inform their learning and practice (explored in particular by [6,7,10 and 11]).

The second theme examines the relationship between computational thinking and other topics in the curriculum. This focus is instantiated in two ways, with de Paula et al. [3] examining projects which necessitate a combination of computation and other non-STEM topics (in this case, storytelling through game development), while Benton et al. [4] consider the the extent to which learning to code allows students to engage with concepts from other domains (mathematics) in new and effective ways.

The third theme considers assessment, which has been a key focus and concern since Wing's consideration of computational thinking in 2016. Papavlasopoulou et al. [5] used a novel approach to assess children's learning behaviours through eye-tracking and correlate this with attitudes towards computing, while Roman-Gonzalez et al. [8] developed a test which allows 'computationally talented' students to be identified even before they start to learn programming.

Finally, the fourth important theme to emerge focuses on the broader context of teaching children about coding and computation and, in particular, the need to support teachers in developing both their confidence and their pedagogical approaches to computing education ([2, 9] and also considered by [4]).

References

1. Howland K., Good J., Robertson, J. and Manches A. Every child a coder?: Research challenges for a 5-18 programming curriculum. In Proceedings of the 14th International Conference on Interaction Design and Children, 2015, Jun 21, pp. 470-473. ACM.
2. Chalmers, C. Robotics and computational thinking in primary school. International Journal of Child-Computer Interaction, Volume 17, 2018, pp. 93-100.

3. de Paula BH, Burn A, Noss R, Valente JA. Playing Beowulf: Bridging computational thinking, arts and literature through game-making. International Journal of Child-Computer Interaction, Volume 16, 2018, pp. 39-46.
4. Benton, L., Saunders, P., Kalas, I., Hoyles, C. and Noss, R. Designing for learning mathematics through programming: A case study of pupils engaging with place value, International Journal of Child-Computer Interaction, Volume 16, 2018, pp. 68-76.
5. Papavlasopoulou, S., Sharma, K. and Giannakos, M. How do you feel about learning to code? Investigating the effect of children's attitudes towards coding using eye-tracking, International Journal of Child-Computer Interaction, Volume 17, 2018, pp. 50-60,
6. Weintrop, D. and Wilensky, U. How block-based, text-based, and hybrid block/text modalities shape novice programming practices, International Journal of Child-Computer Interaction, Volume 17, 2018, pp. 83-92.
7. Gomes, T. C. S., Falcão, T. P., and Tedesco, P. C. D. A. R. Exploring an approach based on digital games for teaching programming concepts to young children. International Journal of Child-Computer Interaction, Volume 16, 2018, pp. 77-84.
8. Román-González M., Pérez-González J.C., Moreno-León J. and Robles G. Can computational talent be detected? Predictive validity of the Computational Thinking Test. International Journal of Child-Computer Interaction. Volume X, 2018.
9. Katterfeldt, E., Cukurova, M., Spikol, D and Cuartielles, D. Physical computing with plug-and-play toolkits:Key recommendations for collaborative learning implementations, International Journal of Child-Computer Interaction, Volume 17, 2018, pp. 72-82.
10. Tsan, J., Lynch, C.F. and Boyer, K.E. 'Alright, what do we need?": A study of young coders' collaborative dialogue, International Journal of Child-Computer Interaction, Volume 17, 2018, pp. 61-71.
11. Kelly, A., Finch L., Bolles M. and Shapiro R.B.. BlockyTalky: New programmable tools to enable students' learning networks. International Journal of Child-Computer Interaction. Volume X, 2018.